Towards Efficient Hardware Debugging using Parameterized FPGA Reconfiguration

Alexandra Kourfali and Dirk Stroobandt
Department of Electronics and Information Systems (ELIS), Ghent University
iGent, Technologiepark-Zwijnaarde 15, 9052 Ghent, Belgium.
{Alexandra.Kourfali, Dirk Stroobandt}@UGent.be

A. Introduction

Ensuring a design's functional correctness within time-to-market constraints, continues to stand as one of the biggest challenges for today's ASIC design teams. Lately, circuit designers have turned to Field-Programmable Gate Arrays (FPGAs) for the simulation of their complete systems (FPGA emulation), that offer operating frequencies that are several orders of magnitude faster than simulation. However, debugging through FPGA emulation has its own challenges, such as lack of on-chip signal observability. Moreover, observing a new subset of signals requires recompilation of the circuit. Each instrument-compile-debug iteration can take multiple hours.

B. Parameterised debugging flow

We propose a low overhead debugging flow, which is entirely automated and integrated within the normal FPGA CAD flow. It offers debug acceleration and enhanced internal signal visibility by using parameterised configurations (PConf). This will allow us to implement parameterised hardware systems, with parameters that define different circuit instances that can be optimised on the fly by reconfiguring for a current set of parameter values. Our proposed debugging flow follows the typical stages of the FPGA CAD flow and consists of two phases: the offline and the online phase.

During the *offline phase* a generalised configuration is created. The circuit is fully compiled, the mapping is fixed and the extra instrumentation is added. This allows all available signals to be connected with trace-buffer IP. In more detail, during the first stage, *Signal Parameterisation*, all signals that can be used for debugging are automatically selected and parameterised. These signals are annotated as parameters, as they will change (but less frequently than the other signals) depending on the set of signals that will be traced during debugging. These parameters can hence be implemented in the reconfiguration resources.[1]

After the extra instrumentation has been added, the design is synthesised and then, during *technology mapping (TCON-Map)*, the parameterized network (generated during synthesis), is not directly mapped onto the resource primitives available in the target FPGA architecture, but intermediately on abstract primitives that introduce and allow the reconfigurability of the logic and routing resources. Finally, the *Place & Route (TPaR)* stage can enable routing of the circuit where its routing resources can be reused during the fault emulation and this drastically reduces the area usage. At the end of the computationally intensive compile phase, a virtual intermediate FPGA configuration is created.

During the *online phase*, for every debugging cycle the network is partially reconfigured with the exact signals the designer wishes to trace at that specific instance. Here, only the configuration cells of all the routing switch boxes and the connection boxes for the memory resources will be reprogrammed, instead of the full recompilation and/or reconfiguration, as it is the case in related work. This online step will be future work, as we first focus on the offline part of the toolflow.

C. Preliminary results

In order to evaluate our proposed method, we are integrating our techniques inside VTR, so that it can be complient with our parameterised debugging infrastructure. The first experiments with the VTR benchmarks indicate that we only need the area for the largest circuit instance implementation, instead of the sum of areas of the initial and the added implementation. This enables us to include debugging infrastructure without area overhead.

Our technique reduces the critical path delay of the added functionality for the faults by reducing the number of LUTs and the routing infrastructure on the critical path. The logic depth (inversely related to clock speed) of the design did not change by adding the extra debugging infrastructure.

The runtime overhead depends on the number of times the emulator needs to be reconfigured and on the time to evaluate the PConf and to reconfigure the bits that changed. The time overhead can be expressed as the single specialization time (for specializing the FPGA once) multiplied by the number of times a new signal set will be activated. The evaluation time is used to evaluate the Boolean functions in the parameterized configuration produced by the offline generic stage of the TCON tool flow (maximum 50 μs). Thus, each parameterised configuration can be 3 orders of magnitude faster than a full reconfiguration (176 milliseconds for a Xilinx Virtex-5 FPGA).

D. Conclusion

A low overhead debugging method is proposed. The main (parameterised) debugging infrastructure is presented, which is meant for both emulation approaches (for ASIC verification)

[1] We assume that the higher utilization of routing resources does not affect the placement of the design.

and on-line in field debugging approaches (for FPGA design verification) and it includes increasing design observability. This infrastructure lies within the circuit implementation and is only invoked when a debugging parameter is set. Therefore, this infrastructure is always present but does not require much additional area. The area needed is found by introducing parameterized reconfiguration in the application. Hence, thanks to the fact that there is virtually no overhead over the original implementation, we can add the debugging functionality for free.